Type-2 Fuzzy Logic Control in Computer Games

Atakan Sahin [1] and Tufan Kumbasar [2]

[1] Centre for Process Analytics and Control Technology, University of Strathclyde, G1 1XL, Glasgow, United Kingdom, Email: atakan.sahin@strath.ac.uk

[2] Control and Automation Engineering Department, Istanbul Technical University, 34469, Istanbul, Turkey, Email: kumbasart@itu.edu.tr [2]

Summary. In this chapter, we will present the novel applications of the Interval Type-2 (IT2) Fuzzy Logic Controllers (FLCs) into the research area of computer games. In this context, we will handle two popular computer games called Flappy Bird and Lunar Lander. From a control engineering point of view, the game Flappy Bird can be seen as a classical obstacle avoidance while Lunar Lander as a position control problem. Both games inherent high level of uncertainties and randomness which are the main challenges of the game for a player. Thus, these two games can be seen as challenging testbeds for benchmarking IT2-FLCs as they provide dynamic and competitive elements that are similar to real-world control engineering problems. As the game player can be considered as the main controller in a feedback loop, we will construct an intelligent control systems composed of three main subsystems: reference generator, the main controller, and game dynamics. In this chapter, we will design and then employ an IT2-FLC as the main controller in a feedback loop such that to have a satisfactory game performance while be able to handle the various uncertainties of the games. In this context, we will briefly present the general structure and the design methods of two IT2-FLCs which are the Single Input and the Double Input IT2-FLCs. We will show that an IT2 fuzzy control structure is capable to handle the uncertainties caused by the nature of the games by presenting both simulations and real-time game results in comparison with its Type-1 and conventional counterparts. We believe that the presented design methodology and results will provide a bridge for a wider deployment of Type-2 fuzzy logic in the area of the computer games.

Keywords: Type-2 Fuzzy Logic; Type-2 Fuzzy Sets; Type-2 Fuzzy Logic Controllers, Computer Games; Games; Lunar Lander; Flappy Bird

1. Introduction

Computer game industry is one of the biggest high-tech industry as well as its revenue. Depending on their virtual worlds which inspire from real-world dynamicity or facts, they are a perfect test-bed for computational intelligence methods or several types of research [1]. These research areas can show extremely diversity depending on their field. For instance, self-awareness of the game bots is the challenging application of computational intelligence under computer science area [2], [3], [4]. On the opposite side, engineers try to build a perfect player to bear against game environment [5], [6], [7]. Furthermore, collected behavior logs of the human players from their plays might also be a

source for social scientists [8]. Consequently, several games have been used as test beds such as Pacman [5], Scrabble [6], Super Mario [9], Counter-Strike [2], StarCraft [10], Flappy Bird [11], [12] and, Lunar Lander [13], [14].

In the last decade, Type-2 (T2) Fuzzy Logic, which is a generalization of ordinary (Type-1) fuzzy logic, have made a significant breakthrough in the area of computational intelligence [15], [16]. Especially, Interval T2 (IT2) Fuzzy Logic Controllers (IT2-FLCs) have been successfully employed in various engineering problems [17], [18] [19], [20], [21], [22], [23]. IT2-FLCs have the capability of handling high-level uncertainties as well as nonlinear dynamics in comparison with its Type-1 (T1) and conventional counterparts. This lies due to the extra degree of freedom provided by their T2 fuzzy sets (T2-FSs) [24], [25]. The superiority of IT2-FLCs has been shown in various control engineering applications such as mobile robots [17], [18], [19], [20], unmanned flight systems [21], engine control [22]. Although the mainstream of these researches are based on Double Input IT2-FLCs (DIT2-FLCs) [14], [16], [18], [19], [20], [21], [23], [29], it has also been shown in [21], [26], [27] that Single Input IT2-FLCs (SIT2-FLCs) are easy to design and to deploy to real-time control engineering applications.

In this chapter, we will present the novel applications of the IT2-FLCs into the research area of computer games. In this context, we will handle two well-known computer games, namely Flappy Bird [11], [12] and Lunar Lander [13], [14], to show the abilities of the IT2-FLCs. From a control engineering point of view, as the game player can be seen as the main controller in a feedback loop, we will transform the game logic of flappy bird into a reference tracking problem while handling the moon landing problem as a position control problem. Thus, we will construct an intelligent control system composed of three main subsystems: reference generator, the main controller, and game dynamics. In this chapter, we will design and then employ an IT2-FLC as the main controller in a feedback loop such that to have a satisfactory performance and to be able to handle the various uncertainties of the games. In this context, we will briefly present the general structure and the design methods of two IT2-FLCs which are the SIT2-FLC and DIT2-FLC. In this chapter, we will design a SIT2-FLC for the game Flappy Bird while a DIT2-FLC structure for the game moon lander. The IT2-FLCs have been designed and implemented by using the Interval Type-2 Fuzzy Logic Toolbox [28] for Matlab/Simulink. We will examine the performance of both IT2 fuzzy control systems with respect to their control system and game performances, in comparison with its T1 and conventional counterparts, to show that the presented structure can handle the uncertainties caused by the nature of the games much better.

2. Interval Type-2 Fuzzy Logic Controllers

The aim of this section is to present the general structure of the PID type IT2-FLCs by classifying them on the number of input variables. In literature, the most widely used IT2-FLC structures are the SIT2-FLC and DIT2-FLC ones as presented in Fig.1 [27], [29].

The SIT2-FLC structure uses only the error signal (e) as its input and the control signal (u_{SIT2}) as its output as shown in Fig. 1a[27]. The handled SIT2-FLC structure has one input and four output Scaling Factors (SFs). Here, K_E is the input SF that normalize the input to the common interval $[-1, +1]$ in which the membership functions (MFs) of the inputs are defined and is defined as:

$$K_E = \frac{1}{e_{max}} \tag{1}$$

where e_{max} represent the maximum allowed error value. While the IT2 fuzzy output (U) is mapped into the respective actual output (u_{SIT2}) by the output SFs as follows:

$$u_{SIT2} = K_P U + K_I \int U dt + K_D \frac{dU}{dt} \tag{2}$$

where K_p, K_I and K_D are defined as:

$$K_P = K_{P0} K_u \quad K_D = K_{D0} K_U \quad K_I = K_{I0} K_u \tag{3}$$

Here, the output SF K_U is set as K_E^{-1} to rescale the IT2-FIS output while K_{P0}, K_{I0} and K_{D0} are the baseline PID controller gains.

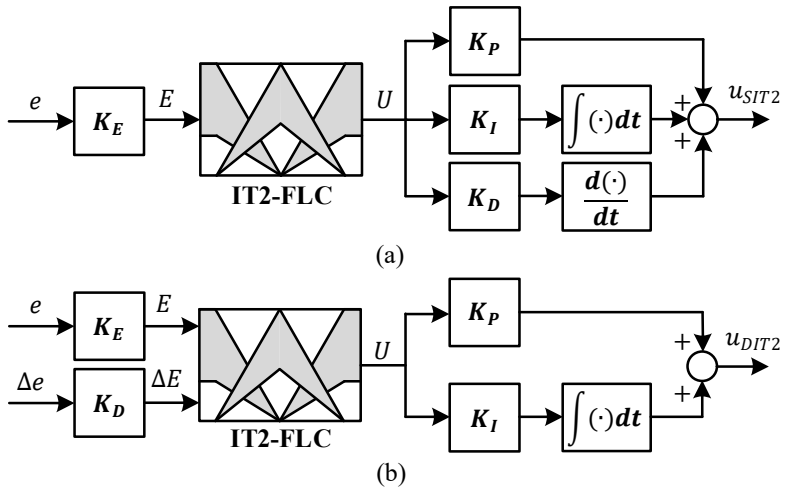

Fig. 1. Illustration of the (a) SIT2-FLC (b) DIT2-FLC structure.

The PID type DIT2-FLC is formed using PD type DIT2-FLC with an integrator and a summation unit at the output as shown in Fig.1b [23], [29]. The PID type DIT2-FLC given in Fig. 1b is constructed by choosing the inputs to be an error (e) and change of the error (Δe) and the output as the control signal (u). Here, the input SFs K_E (for e) and K_d (for Δe) normalize the inputs to the universe of discourse where the MFs of the inputs are defined. Thus, e and Δe are converted after normalization into E and ΔE while the output (U) of the PID type DIT2-FLC is converted into the control signal (u) by the output SFs K_P (proportional SF) and K_I (integral SF) as follows:

$$u_{DIT2} = K_P U + K_I \int U dt \tag{4}$$

It can be concluded that both the SIT2-FLC and DIT2-FLC controllers are analogous to the conventional PID controllers from the input-output relationship point of view [23], [27], [29]. The main difference of these structure lies in the characteristics of their internal structure.

2.1 Single Input Interval Type-2 Fuzzy Logic Controller

In this subsection, we will present the internal structure of the SIT2-FLC and the key factors in its design. The rule structure of the SIT2-FLC is as follows:

$$R_q\text{: IF } E \text{ is } \tilde{A}_{1j} \text{ THEN } U \text{ is } B_j \qquad j = 1, 2, 3 \tag{5}$$

where B_l are the crisp consequents with description on these as $B_1 = -1$, $B_2 = 0$ and, $B_3 = 1$ with rule index $q = 1, 2, 3$. The antecedent MFs are defined with triangular IT2-FSs $(\tilde{A}_{1j})$ as represented in Fig. 2 and defined with three linguistic terms: Negative (N), Zero (Z), Positive (P). The IT2-FSs are described with an upper MF (UMF) $\overline{\mu}_{\tilde{A}_{1j}}$ and Lower MF (LMF) $\underline{\mu}_{\tilde{A}_{1j}}$ that provide an extra degree of freedom named as Footprint of Uncertainty (FOU) [27], [29]. As shown in Fig.2, m_{ij}'s represent the height of the LMFs and will be the main design parameters of the SIT2-FLC to be tuned. For the sake of simplicity, we employ $m_{i2} = \alpha$ and $m_{i1} = m_{i3} = 1 - \alpha$. Thus, α is the new design parameter which determines the FOU in the antecedent IT2-FSs [27].

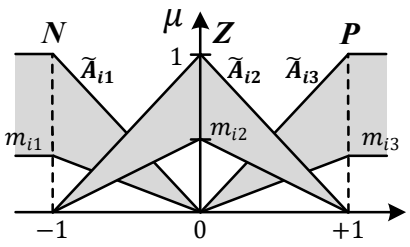

Fig. 2. Illustration of the antecedent of the IT2-FS.

The implemented the SIT2-FLC uses the center of sets type reduction method [21], [26], [27]. It has been demonstrated in [27] that the defuzzified output can be calculated as:

$$U = \frac{U_l + U_r}{2} \tag{6}$$

where U_l and U_r are the left and right end points respectively of the type reduced set, are defined as follows:

$$U_l = \frac{\sum_{q=1}^{R} \underline{\mu}_{\tilde{A}_{1q}} B_q + \sum_{q=R+1}^{N} \overline{\mu}_{\tilde{A}_{1q}} B_q}{\sum_{q=1}^{R} \underline{\mu}_{\tilde{A}_{1q}} + \sum_{q=R+1}^{N} \overline{\mu}_{\tilde{A}_{1q}}} \tag{7}$$

$$U_r = \frac{\sum_{q=1}^{L} \overline{\mu}_{\tilde{A}_{1q}} B_q + \sum_{q=L+1}^{N} \underline{\mu}_{\tilde{A}_{1q}} B_q}{\sum_{q=1}^{L} \overline{\mu}_{\tilde{A}_{1q}} + \sum_{q=L+1}^{N} \underline{\mu}_{\tilde{A}_{1q}}} \tag{8}$$

where R and L are the switching points [27]. As shown in Fig.2, the SIT2-FLC employs fully overlapping IT2-FSs in the sense of LMFs and UMFs. Hence, it is guaranteed that a crisp value E' always belongs to two successive IT2-FSs $(\tilde{A}_{1j}$ & $\tilde{A}_{1j+1})$. Thus, since only $N = 2$ rules will be always activated, the values of R and L are equal to 1 [27]. Moreover, the input-output mapping of the SIT2-FLC $(U(E))$ can be derived as follows [26], [27]:

$$U(E) = E \cdot k(|E|) \tag{9}$$

where, $k(E)$ is the T2 fuzzy gain and defines as:

$$k(E) = \frac{1}{2}\left(\frac{1}{\alpha + E - \alpha E} + \frac{\alpha - 1}{\alpha E - 1}\right) \tag{10}$$

Accordingly, the main design parameter of the IT2 FSs, α, is assigned as the only tuning parameter. This derivation simplifies the SIT2-FLC design into a Control Curve (CC) generation instead of conventional control surface design. In [27], by defining $\varepsilon_0(E) \ = U(E) - E$, the following design guidelines have been presented.

- If $0 < \alpha \leq \alpha_{c1}$, then $\varepsilon_0 < 0$ for $\forall E \in \mathrm{O}_S$ where $\mathrm{O}_S \in [0,1)$ and $\alpha_{c1} = \left(3 - \sqrt{5}\right)/2$. Thus, a Smooth $\mathrm{CC}_{\mathrm{IT2}}$ $(\mathrm{S} - \mathrm{CC}_{\mathrm{IT2}})$ will be generated.
- If $\alpha_{c2} \leq \alpha < 1$, then $\varepsilon_0 > 0$ for $\forall E \in \mathrm{O}_A$ where $\mathrm{O}_A \in [0,1)$ and $\alpha_{c2} = \left(\sqrt{5} - 1\right)/2$. Thus, an Aggressive $\mathrm{CC}_{\mathrm{IT2}}$ $(\mathrm{A} - \mathrm{CC}_{\mathrm{IT2}})$ will be generated.

In Fig. 3, a $\mathrm{S} - \mathrm{CC}_{\mathrm{IT2}}$ and $\mathrm{A} - \mathrm{CC}_{\mathrm{IT2}}$ examples are given for $\alpha = 0.2$ and $\alpha = 0.8$, respectively. Here, a Unit CC $(\mathrm{U} - \mathrm{CC})$ is sketched for the comparison. It can be clearly seen that the $\mathrm{S} - \mathrm{CC}_{\mathrm{IT2}}$ has relatively low input sensitivity when E is close to "0" when compared to the $\mathrm{A} - \mathrm{CC}_{\mathrm{IT2}}$. Thus, the parameter α of the SIT2-FLC can be tuned such that to enhance the control performance of its baseline counterpart via the design guidelines [27].

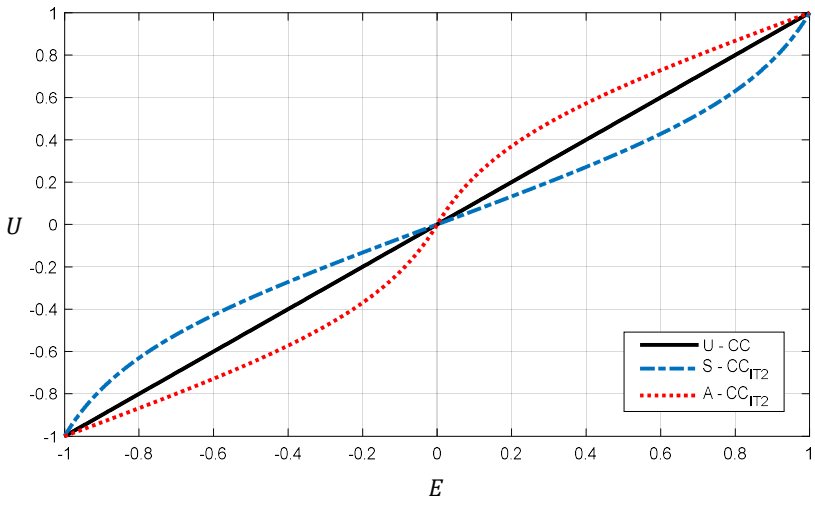

Fig. 3. Illustration of the $\mathrm{CC}_{\mathrm{IT2}}s$.

2.2 Double Input Interval Type-2 Fuzzy Controller

In this subsection, we will present the internal structure of the DIT2-FLCs. The DIT2-FLC, handled in this study, uses and employs the $3x3$ rule base given in Table.1. The rule structure is defined as follows:

$$R_q\text{: IF } E \textit{ is } \tilde{A}_{1i} \text{ and } \Delta E \textit{ is } \tilde{A}_{2j} \text{ THEN } U \text{ is } C_q \tag{11}$$

where C_q is the crisp consequent MFs $(q = 1, \dots, Q = 9)$ is defined five linguistic terms Negative Big (NB), N, Z, P and Positive Big (PB) that represent $-1, -0.5, 0, 0.5, 1$, respectively. The antecedent part of the rule is defined with

IT2-FSs $\left(\tilde{A}_{1i}, \tilde{A}_{2j};\ i = 1,2,3; j = 1,2,3\right)$ which are defined with three linguistic terms N, Z and P. The IT2-FSs can be described with UMFs ($\overline{\mu}_{\tilde{A}_{1i}}$ and $\overline{\mu}_{\tilde{A}_{2j}}$) and LMFs ($\underline{\mu}_{\tilde{A}_{1i}}$ and $\underline{\mu}_{\tilde{A}_{2j}}$) which provides extra degree of freedom that is also known as FOU. Similar to its input counterpart, the FOU of IT2-FSs is generated with the heights of the LMFs $\left(m_{ij}\right)$ which is the only design parameter to be tuned [23], [29].

Table 1. Rule table.

$\Delta E/E$	N	Z	PB
N	NB	N	Z
Z	N	Z	P
P	Z	P	PB

The implemented DIT2-FLC uses the center of the sets type reduction method [29]. It has been demonstrated in [29] that the defuzzified output can be calculated as:

$$U = \frac{U_l + U_r}{2} \tag{12}$$

where U_l and U_r are the left and right end points respectively of the type reduced set, are defined as follows:

$$U_l = \frac{\sum_{q=1}^{L} \overline{f}_q C_q + \sum_{q=L+1}^{Q=9} \underline{f_q} C_q}{\sum_{q=1}^{L} \overline{f}_q + \sum_{q=L+1}^{Q=9} \underline{f_q}} \tag{13}$$

$$U_r = \frac{\sum_{q=1}^{R} \underline{f_q} C_q + \sum_{q=R+1}^{Q=9} \overline{f}_q C_q}{\sum_{q=1}^{R} \underline{f_q} + \sum_{q=R+1}^{Q=9} \overline{f}_q} \tag{14}$$

where R and L are the switching points defined between $[1, Q-1]$ [15-17]. Moreover, $\tilde{f}_q = \left[\overline{f}_q \quad \underline{f_q}\right]$ is the total firing strength for the q^{th} rule and is defined as:

$$\underline{f_q} = \underline{\mu}_{\tilde{A}_{1i}} * \underline{\mu}_{\tilde{A}_{2j}} \tag{15}$$

$$\overline{f}_q = \overline{\mu}_{\tilde{A}_{1i}} * \overline{\mu}_{2j} \tag{16}$$

Here, " $*$ " represents the product implication (the t-norm). The typed reduced set can be calculated by finding the optimal switching points (R and L) with the Karnik and Mendel algorithm [15], [16], [29].

In control engineering applications, it is usually desired to design a symmetric control surface [20],[27],[29], [30]. In [31], [32], it has been shown that, by setting $m_{11} = m_{13}$ and $m_{21} = m_{23}$, how to generate smooth and aggressive control surfaces by simply tuning the FOU parameters. In Fig. 4a, a smooth control surface is presented for the FOU parameter settings $m_{11} = m_{13} = 0.3$, $m_{12} = 0.9$, $m_{21} = m_{23} = 0.3$ and $m_{22} = 0.9$ while in Fig. 4b an

aggressive control surface is presented for the FOU parameter settings $m_{11} = m_{13} = 0.9, m_{12} = 0.1, m_{21} = m_{23} = 0.9$ and $m_{22} = 0.1$.

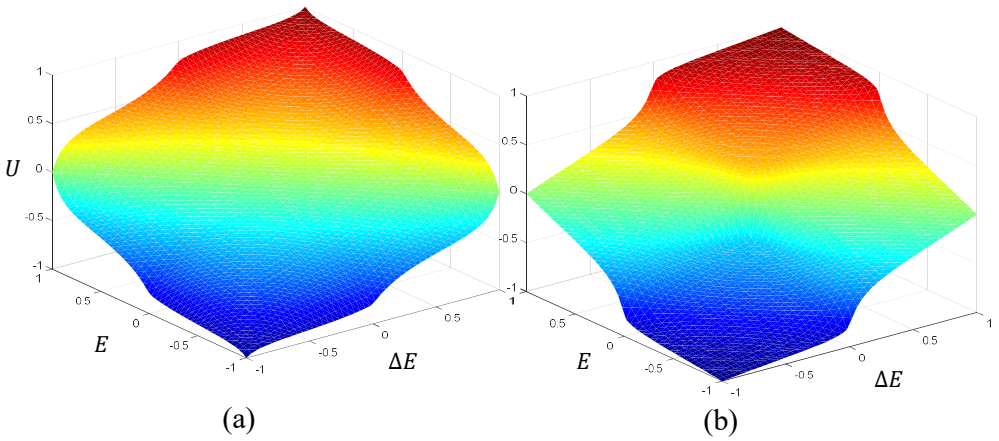

Fig. 4. Illustration of the (a) smooth, (b) aggressive control surface of DI-IT2 FLCs.

3. Type-2 Fuzzified Flappy Bird Control System

In this section, we will represent the design and performance evaluation of the T2 Fuzzified Flappy Bird Control System for the game Flappy Bird. Flappy Bird is published as a mobile game in 2013 by GEARS Studios [33]. As a side-scrolled game, the basic game logic of the game is as follows: each time the player taps the screen, the bird flaps its wings, moving upward in an arc while gravity constantly pulls downward; if the screen is not tapped, the bird falls to the ground due to gravity, which also ends the game. The main goal of the game is to control the bird's height while attempting to fly between the obstacles (i.e. the pipes) without hitting them [11], [33].

3.1. Game Space

Here, we will briefly explain the game space of the Matlab replica of the game Flappy Bird that can be downloaded from [34]. In this version of this game, the game parameters are grouped as the parameters of the environment $(p_g, \overline{p}_h, \underline{p}_h, p_s, p_w, H)$ and the bird's dynamics (g, v, u). These parameters are defined as follows with their illustration in Fig.5 [11], [33]:

- **World height** (H) is the distance between the ceiling and the floor with a fixed value of 180 pixels.
- **Upper pipe height** $(\overline{p}_h)$ is the distance between the top of the upper pipe and ceiling. This value is generated by a uniformly distributed random number generator.
- **Pipe gap** (p_g) is the distance between the pipes which is fixed as 49 pixels.
- **Lower pipe height** $(\underline{p}_h)$ is the distance between the top of the lower pipe and floor. The value of the lower pipe height is defined as $\underline{p}_h = \overline{p}_h - 49$. Though, it is constrained with a minimum value as 36 pixels.

- **Pipe width** (p_w) is the width of the pipe with a fixed value of 24 pixels.
- **Pipe separation** (p_s) is the horizontal space between two consequent pipes with a fixed value of 80 pixels.
- **Gravitational constant** (g) has a default value assigned as 0.1356 pixels per frame.
- **Bird's x-direction velocity** (v_x) has a fixed value of 1 pixel per frame.
- **Control signal** (u) is the binary input variable $u \in \{0,1\}$ provided by the user to flap the bird on the y direction velocity (v_y) [11].

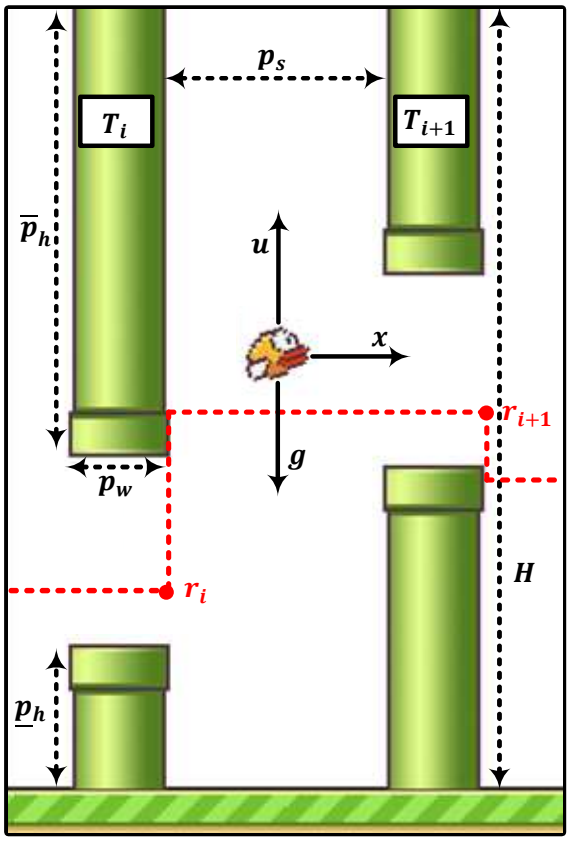

Fig. 5. Illustration of the game space and one of the sample generated reference for Flappy Bird.

3.2. The Intelligent Control System for Flappy Bird

In this subsection, we will use the presented game space of Flappy Bird and convert the obstacle avoidance problem into a reference tracking control problem. The proposed T2 fuzzy control system is composed of three main parts which are the reference generator, the SIT2-FLC, and the system dynamics of the bird as illustrated in Fig. 6. We will handle the bird as the dynamic system to be controlled, the pipe gap as the goal to be tracked via the reference trajectory and the environment generation as uncertainty and disturbance [12].

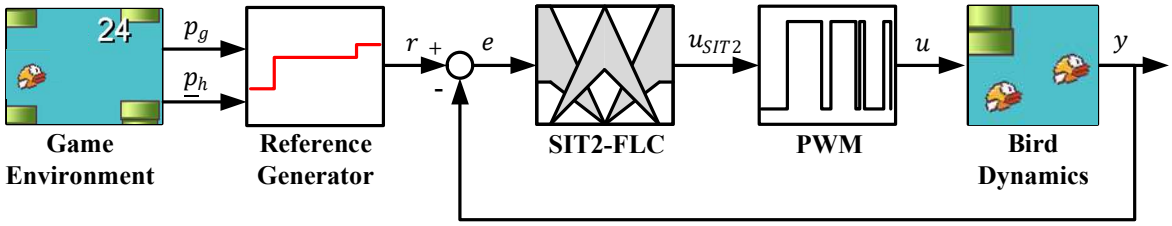

Fig. 6. Illustration of the T2 Fuzzified Flappy Bird Control System.

3.2.1. The System Dynamics of Flappy Bird

As it has been asserted in the preceding section, the bird has a constant horizontal (x) velocity while the vertical (y) velocity depends on the player's taps which directly controls the dynamics of the bird. From a control engineering point of view, the taps can be seen as the control signal of the system which is based on binary numbers and the vertical velocity to be controlled. In the rest of the section, we will use the abbreviation v for representing v_y since v_x is constant variable as described in the game logic. The bird's system dynamics are defined as follows [34]:

$$y_t = y_{t-1} + v_t$$
$$v_t = \begin{cases} 2.5, & u = 1 \\ v_{t-1} - g, & u = 0 \end{cases} \tag{17}$$

where v_t and y_t are vertical velocity and vertical position of the bird at t^{th} frame as respectively.

3.2.2. The Reference Generator

The reference generator is an essential component in the control loop as it transforms the obstacle avoidance problem of the game to a fuzzy feedback control system. The reference generator provides the trajectory for the bird by taking account the gap between the pipes and the bird's position. The reference trajectory (r_i) is updated when the bird reaches the end of the pipe set (T_i) automatically as shown in Fig.5 (the red line). The new reference trajectory is defined as:

$$r_{i+1} = \underline{p}_h + 0.3\left(\overline{p}_h - \underline{p}_h\right) \tag{18}$$

where i is the frame when the bird reaches the end of the pipe.

3.2.3. The Interval Type-2 Fuzzy Logic Controller Structure

For the presented fuzzy control system in Fig.6, we will prefer a SIT2-FLC structure given in Fig.1a. Furthermore, as it is crucial not to hit and not to track the reference signal with zero steady state error, we will prefer to employ a P type SIT2-FLC for the sake of simplicity ($K_{I0} = 0, K_{D0} = 0$). In this structure, we will set and fix the input SF K_E as $K_E = 1/150$ while the output SF K_{P0} will be set and fixed to its baseline counterpart (its design will be explained in Section 3.3). Thus, the only parameter to be tuned in the SIT2-FLC structure is the FOU design parameter α. It is worth to note that the generated signal from the P type SIT2-FLC needs also to be converted a binary signal. Thus, the continuous control signal (u_{IT2}) is then converted into a Pulse Width Modulation (PWM) generator, which is widely used in power electronics [35], into the input signal $u \in \{0,1\}$.

3.3. The Design and Performance Evaluation of the SIT2-FLC structure

This subsection will include the design steps of the SIT2-FLC and then investigate its performance in comparison with the conventional P controller. Then, we will present experimental results that performed in both the simulation and game environment to examine its game performance.

As it has been asserted in Section 2.1, the design of the SIT2-FLC is accomplished as an extension of its baseline counterpart. Thus, as it has been preferred to design and employ a P type SIT2-FLC, it is necessary to design a conventional P controller. In this context, the Genetic Algorithm is used to find the optimal proportional gain value K_{P0} on a randomly generated training reference trajectory regarding the game logic as illustrated in Fig.7a. The defined cost function is the widely used Integral of Time Absolute Error (ITSE) which is given below:

$$\text{ITSE} = \int e^2(t)t\,dt \tag{19}$$

The resulting optimal parameter value for the proportional gain is found as $K_{p0} = 5.04$ from the training trajectory. Then, to enhance the control performance of its baseline counterpart in presence of uncertainties and nonlinearities, we preferred to set the FOU design parameters value as $\alpha = 0.2$ to end up with $\mathrm{S-CC_{IT2}}$ which is potentially more robust controller in comparison with its baseline and $\mathrm{A-CC_{IT2}}$ counterparts. The control system performances of the SIT2-FLC and P controller structures are given in Fig.7a, and their corresponding ITSE values are 30130 and 31630, respectively. It can be concluded that the SIT2-FLC reduced the ITSE value by about 5% in comparison to its baseline counterpart in the training phase. Moreover, since the dynamics of flappy bird system inherent nonlinearity as given in (17), we have examined the controller performances for a testing trajectory which is also generated randomly as shown in Fig. 7b. In other words, we have tested the controller performances for different operating points at which they have not been designed. The ITSE values of the SIT2-FLC and P controller structures are calculated as 99674 and 136270, respectively. Thus, in comparison with its conventional counterpart, the SIT2-FLC resulted with a better tracking performance as shown in Fig. 7b and was also able to reduce the ITSE value by about 27% on the testing trajectory. Moreover, with respect to the game logic; it is worth to underline at the reference variation (r_i) in the 1200^{th} frame (the shaded area in Fig. 7b) that the conventional control system almost hit the T_{i+1} pipe which would end the game. On the other hand, for the same operation point, the T2 fuzzified flappy bird control system resulted with a satisfactory reference tracking performance.

In the real game environment, the ITSE value comparison loses its importance as the number of successfully avoided pipes is the indicator of the score rather than the reference tracking performance. Therefore, since the game environment parameters such as pipe gap's location $\left(\overline{p}_h, \underline{p}_h\right)$ are generated randomly during the game, we have repeated each experiment 20 times to get an overall performance comparison. The results of the game performances are tabulated in Table II where the best and average scores of the experiments are given. It can be clearly observed that the T2 fuzzy control scheme improved the average score almost by 55% in comparison with its baseline counterpart. Consequently, the SIT2-FLC structure is better when compared to its conventional counterpart with respect to both its control and game performance.

Table 2. Control performance evaluation for Flappy Bird.

	Average	*Best*
P Controller	112	423
SIT2-FLC	174	482

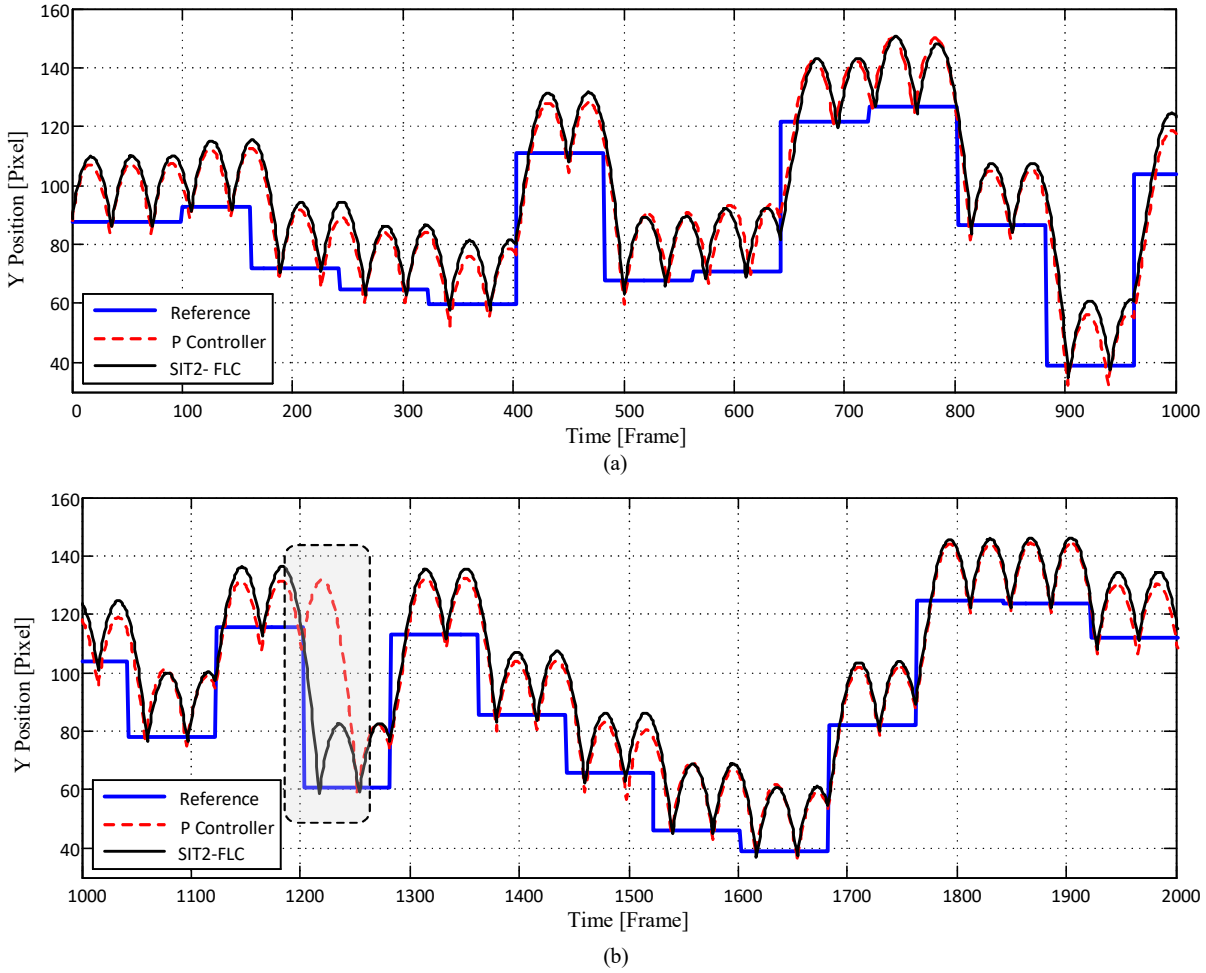

Fig. 7. System responses of the feedback control systems for the (a) training reference trajectory (b) testing reference trajectory.

4. Type-2 Fuzzy Moon Landing System

In this section, we will represent the design and performance evaluation of the T2 Fuzzy Moon Landing System for the game Lunar Lander. Lunar Lander is one of the most cloned games [36] into several platforms which is a vector monitor based arcade game firstly released by Atari in the late 1970s. The game logic of the Lunar Lander game is to control the engine of the spaceship in the x-y coordinate system such that to land on the dock softly [37,38]. The player can arrange the spaceship's angular rotation by pressing the right or left arrow keys on a cumulative basis. The player can

also produce thrust against the gravity by pushing the spacebar key. The direction of the force obviously depends on the spaceship's angular position [12], [13].

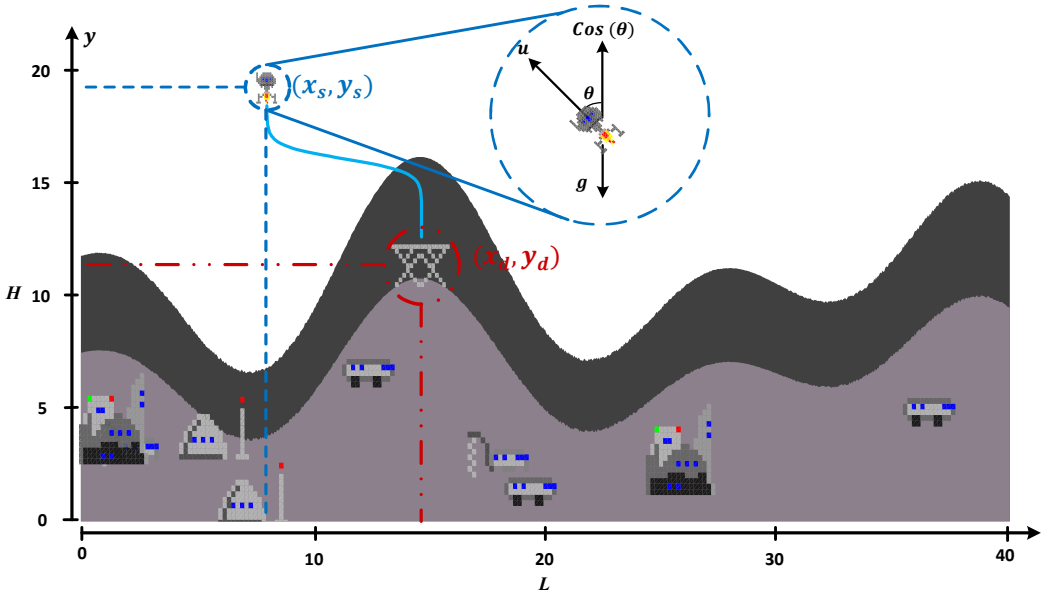

Fig. 8. Illustration of the game environment of Lunar Lander.

4.1. Game Space

In this study, we will use the Matlab clone of the Lunar Lander that can be found in [39]. The game parameters of the game space are illustrated in Fig.8 and defined as follows

- **World height** (H) is fixed to a value of 20 units.
- **World length** (L) is fixed value of 40 units.
- **The terrain** is defined and generated with randomly generated sinus functions. In Fig.8, light gray basements define the basic terrain, which is also obstacles to be avoided by the player. On the other hand, the dark gray ones only provide a realistic game environment, thus are not obstacles.
- **Position of the Dock** (x_d, y_d) is the reference point or the target position for landing on x-y coordinates. The x-axis is generated as randomly while the y-axis depends on the randomly generated light gray terrain.
- **Position of the Spaceship** (x_s, y_s) is the position of the spaceship on x-y coordinate. In the start of the game, the x-axis position of the spaceship is randomly assigned in the range of the world length (L) while its y-axis position is set to a 20 units.
- **Speed of the Spaceship** (v_x, y_y) defines the linear speed of spaceship in x-y the coordinate system.
- **Gravitational constant** (g) has a default value assigned as 0.4 units per frame.
- **The angular position of the Spaceship** (θ) is the angle between the spaceship's direction and y-axis. The angle can be controlled with right and left arrow keys by player.

- **Thrust power** (T) defines the thrust force to be employed to the spaceship engine and is controlled by the player.
- **Fuel** $\left(\int T\right)$ defines the maximum total thrust power that can be consumed by player.

In this Matlab clone [39], the game ends when

1) The spaceships touches/ hits the terrain with failure
2) The spaceship consumes more than 200-unit fuel before a successful landing with failure
3) The Euclidean distance between the spaceship and the dock is less than 2 units with successful landing

Moreover, we have added the following conditions for a successful landing to make the game more realistic with providing soft landing conditions:

4) Vertical velocity (v_y) must be less than -0.5 when the spaceship has landed.
5) The angular position of the spaceship (θ) must be between $[-\pi/16, \pi/16]$ when the spaceship has landed.

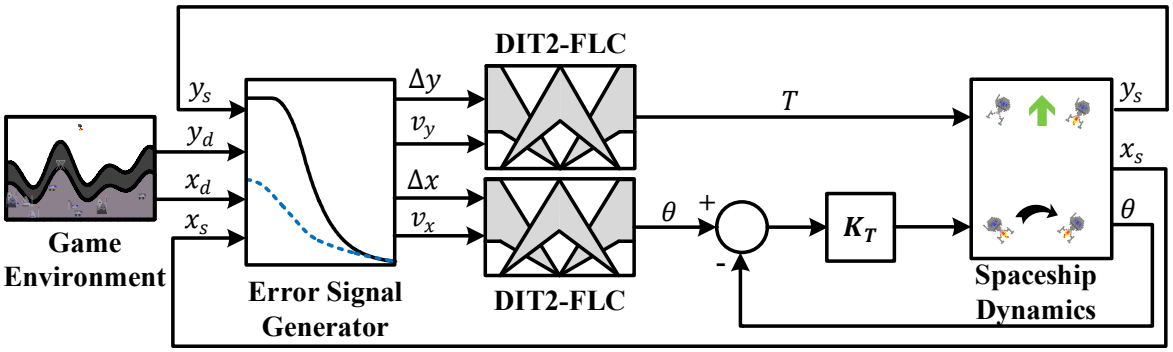

Fig. 9. Illustration of the Type-2 Fuzzy Moon Landing System.

4.2. The Intelligent Control System for Lunar Lander

In this section, we will convert the defined game space of Lunar Lander into a feedback control problem. Here, the spaceship will define as the system to be controlled and the position of the dock (x_d, y_d) to be the desired reference point. Randomly generated initial spaceship position and terrain, and the gravity will be considered as the disturbances and uncertainties in the control loop. Moreover, error signal generator will convert the position signals from the dock (x_d, y_d) and spaceship (x_s, y_s) with game disturbances such as g into feasible reference signal for the controllers that will define to control spaceship's angular position and thrust power via angle (θ) and thrust (T) signals. Therefore, we can define the proposed structure into three part as illustrated in Fig.9. Note that, we have also designed an inner loop proportional controller to speed up the response time of angle of the spaceship (θ). In all experiments, we have set and fixed this controller gain as $K_T = 1.648$.

4.2.1. The System Dynamics

The dynamics of the spaceship are based on the classical motion equations. Thus, the acceleration (a) of the spaceship at kth frame in x-y coordinate system can be defined as:

$$\begin{aligned} a_x(k) &= -\sin\big(\theta(k)\big)\,T(k)d_t \\ a_y(k) &= \quad \cos\big(\theta(k)\big)\,T(k)d_t \end{aligned} \tag{20}$$

where d_t is sampling time of the game with a fixed value of 0.1. Correspondingly, the velocity (v) equations of the both axis can be defined as:

$$\begin{aligned} v_x(k+1) &= v_x(k) + a_x(k)d_t \\ v_y(k+1) &= v_y(k) + a_y(k)d_t - gd_t \end{aligned} \tag{21}$$

Moreover, the position of the spaceship (x_s, y_s) can be defined as:

$$\begin{aligned} x_s(k+1) &= x_s(k) + v_x(k)d_t \\ y_s(k+1) &= \ y_s(k) + v_y(k)d_t \end{aligned} \tag{22}$$

4.2.2. The Error Signal Generator

The error signal generator is designed to transform the landing of the spaceship into a control problem by providing the essential reference signals to the controllers. As the aim of the game is to land the spaceship on the dock, the position differences between the dock and spaceship are used to define the following error signals:

$$\begin{aligned} \Delta x(k) &= x_d - x_s(k) \\ \Delta y(k) &= \ y_d - y_s(k) \end{aligned} \tag{23}$$

Note that, the values x_d and y_d are fixed to the randomly generated values at the beginning of the game until the game ends.

4.2.3. The Interval Type-2 Fuzzy Logic Controller Structure

As it has been asserted, the player has to control the thrust and angle of the spaceship for a successful landing. In this context, we will design two DIT2-FLCs to control the thrust power and angular position of the spaceship to provide a successful landing as illustrated in Fig.9. Here, we preferred a PD type DIT2-FLC structure since according to the success criteria of the game velocity and position must be smaller than predefined values. Note that, we have not preferred a PID structure, as it is necessary to eliminate the steady state error according to the definitions of a successful landing presented in Section 4.1 with 3rd condition for the game ending. As it can be seen from Fig.9, we employed two PD type DIT2-FLC to solve the thrust (T) and angle (θ) control problems of the lunar lander. The DIT2-FLC structure for angle control is constructed by choosing the inputs as $e = \Delta x(k)$, $\Delta e = v_x(k)$ and the output as $u = \theta(k)$. In similar manner, the DIT2-FLC structure for thrust control is constructed with $e = \Delta y(k)$ and $\Delta e = v_y(k)$ and $u = T(k)$. For each DIT2-FLC, there are 2 SFs, excluding the SF K_E , and 6 FOU parameters to be tuned, thus in total $2x8$ parameters for DIT2-FLC structure.

4.3. The Design and Performance Evaluation of the DIT2-FLC structure

This subsection will include the design steps of the DIT2-FLC structure and investigate its performance in comparison with its T1 fuzzy and conventional PD counterparts. Then, to examine the differences on the different level of the uncertainties, experimental results collected from the game environment are presented. The T1-FLC structure is composed with the identical rules of the DIT2-FLC (Table.1) ones with the only difference that it uses and employs triangular type T1-FSs [25].

As it has been mentioned, the Lunar Lander is a limited type game depending on game ending condition and also includes random parameter initializations for each trial. Therefore, the parameter tuning phase of the controllers should be accomplished with several trials in the game space to design controllers that are robust for randomly generated game environments. To provide that, we defined 4 training sets, as tabulated in Table.3, and then tuned the controllers via respectively. The starting position of the spaceship is set and fixed during this phase to the value $(16,20)$. Moreover, the terrain characteristics have also been set and fixed in the training phase to make a fair comparison between the controllers. Here, all three controller structures were optimized with the particle swarm optimization that subject to minimization of the given objective function:

$$F = \sum_{k=1}^{n} \|\boldsymbol{e}_{\boldsymbol{\Delta}}(k)\|^2 + \mathcal{C} \tag{24}$$

where n represents the total number of samples which has been taken for each sampling time starting from the beginning to landing or crash. $\|\cdot\|$ is norm operator for the error term $e_{\Delta}(k)$ which is defined as:

$$\boldsymbol{e}_{\boldsymbol{\Delta}}(k) = [\Delta x, \Delta y]_{(1x2)} \tag{25}$$

Moreover, $\mathcal{C}$ is the penalty for a crash defined as:

$$\mathcal{C} = \begin{cases} 10000, & if\ crash = 1 \\ 0, & if\ crash = 0 \end{cases} \tag{26}$$

Note that, to show the superiority of DIT2-FLCs clearly, we have not optimized the SFs of the DIT2-FLCs. We have set and fixed them to the optimal values found for its T1 counterpart. The resulting optimal parameters are tabulated in Table 3 according to the training scenarios tabulated in Table 4. To provide a further comparison, we have also provided the resulting Landing Times (LT) and the existence of a crash. Firstly, it can be clearly observed that all three control structures achieved their training scenarios without a crash as expected from the training phase. The system responses for Scenario 1 and 4 are illustrated in Fig.10a and Fig.10b, respectively. For Scenario 3, the DIT2-FLC structure decreased the Landing Time value about 41% and 3% (increased the convergence speed to dock) while it also reduced the total fitness value about 30% and 16% in comparison to the PD and T1-FLC structures, respectively. It should be noted that the performances of the T1 and T2 Fuzzy Moon Landing Systems are quite similar. That lies because the design of the DIT2-FLC has been accomplished as an extension of its T1 counterpart. Moreover, this also coincides with the results presented in [20] where it has been stated that the DIT2-FLCs result in smoother control

surfaces in comparison with its T1 counterpart. Thus, the resulting system response might be relatively slower but potentially more robust against uncertainties. Similar comments can be made for the other two reference variations.

Table 3. Controller parameters for Lunar Lander.

Method	*Controller*	*Parameter*	*Value*
PD	Thrust	K_P	5.22
		K_D	13.14
	Angle	K_P	0.51
		K_D	6.12
T1-FLC	Thrust	K_E	$1/e_{max}$
		K_P	39.51
		K_D	0.18
	Angle	K_E	$1/e_{max}$
		K_P	41.37
		K_D	0.61
DIT2-FLC	Thrust	m_{11}, m_{13}	0.27
		m_{12}	0.91
		m_{21}, m_{23}	0.11
		m_{22}	0.67
	Angle	m_{11}, m_{13}	0.15
		m_{12}	0.61
		m_{21}, m_{23}	0.47
		m_{22}	0.91

Table 4. Control performance evaluation of training and testing scenarios for Lunar Lander

				PD			**T1-FLC**			**DIT2-FLC**		
	Scenario	$x_s(0), y_s(0)$	x_d, y_d	F	LT	C	F	LT	C	F	LT	C
Training	1	(16,20)	(2, 5.4)	1612	17.6	N	1508	14.2	N	1520	14.6	N
	2		(10,4.2)	1577	21.2	N	1278	16.7	N	1307	17.0	N
	3		(18,5.6)	1284	19.9	N	1087	12.1	N	905	11.7	N
	4		(25,4.3)	1626	20.7	N	1394	17.2	N	1372	16.5	N
Testing	5	(35,20)	(2, 5.4)	13093	−*	Y	3938	23.5	N	3775	22	N
	6		(5, 3.6)	12932	−*	Y	13107	−*	Y	3500	22.4	N
	7		(10,4.2)	12394	−*	Y	12607	−*	Y	2793	20.4	N
	8		(35,5.5)	1368	20.5	N	10915	−*	Y	1317	20.6	N

***Not applicable because of the crash**

We have also tested the controllers for a different initial point $(35,20)$ to see the performance of the controllers in terms of uncertainties and different operating regions. The resulting performance values are also tabulated in testing

part of Table 4. The system responses for Scenario 5 and 7 are illustrated in Fig.10c and Fig.10d, respectively. It can clearly observe that the T2 fuzzy moon landing system was able to pilot the spaceship to the dock without a crash for all testing scenarios while T1 and conventional PD controller structures crashed the spaceship in three of them. The PD structure was not able to handle the uncertainty and thus, in the first three testing scenarios, the crash occurred for several reasons. Scenario 5 was failed because the 4^{th} condition of the successful landing (presented in Section 4.1) was violated. PD structure has also violated 1^{st} condition at Scenario 6 and 7. The T1 fuzzy structure crashed the spaceship in Scenario Numbers 6 and 7 since it hit/touched the terrain (1^{st} condition). The last crash (Scenario 8) of the T1-FLC structure occurred due to the fact the required angle condition (5^{th} condition) for a successful landing could not be satisfied as it resulted with oscillating system response. The handled scenarios clearly show that the proposed T2 fuzzy moon landing system can handle uncertainties and various operating points when compared to its T1 and conventional PD controller counterparts.

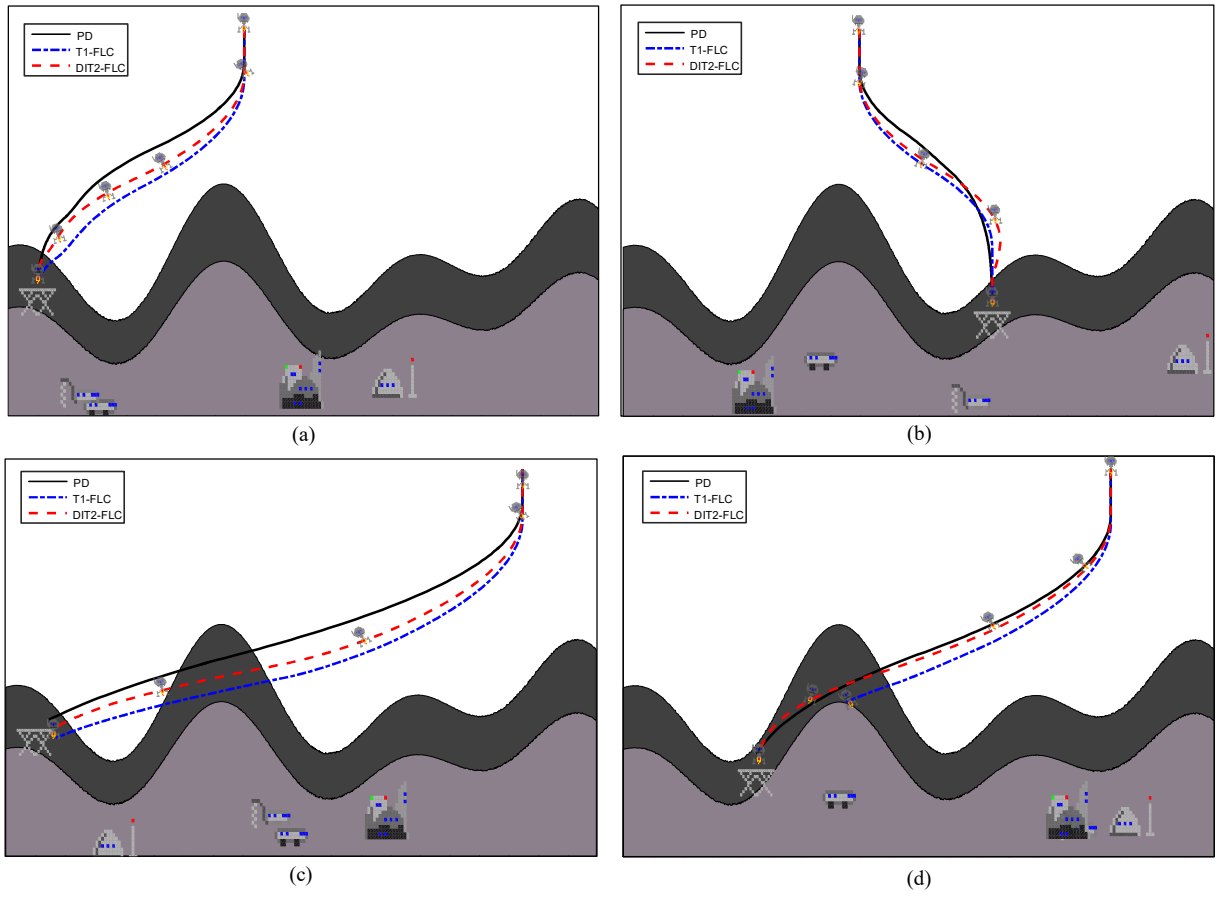

Fig. 10. Illustration of some training and testing cases. (a) Scenario-1 and (b) Scenario-4 for training, (c) Scenario-5 and (d) Scenario-7 for testing.

Distinctly from the simulation studies, the game environment includes various uncertainty sources and operating points caused by its randomization processes in the game logic such as various unique terrains, starting points, and landing points. Therefore, using squared error based evaluation criteria for comparison in the game environment

might not be meaningful. Thus, we will use the successful landing criteria for the testing and compare the controllers in the game environment. In this context, we have employed 200 times each controller structure to game where starting point (x_s, y_s), landing point (x_d, y_d) and also terrain characteristics are randomly generated by the game. The success rates of the controllers are given in Table.5. It can be concluded that the game performance of the DIT2-FLC structure, with respect to game logic, is better than its T1 and conventional parts by almost 14% and 22.5%, respectively.

Table 5. Game performance evaluation on 200 trials for Lunar Lander

	Crash Count	***Success Rate***
PD	78	61%
T1-FLC	61	69.5%
DIT2-FLC	33	83.5%

5. Conclusions

In this chapter, we presented the novel applications of the IT2-FLCs into the well-known computer games called Flappy Bird and Lunar Lander. As these games include various dynamics, achievements and the uncertainties; these two games are as challenging testbeds for benchmarking IT2-FLCs as they provide similar real-world control engineering problems. From a control engineering point of view, as the game player can be seen as the main controller in a feedback loop, we have transformed the game logic of flappy bird into a reference tracking problem while handed the moon landing problem as a position control problem. Then, we proposed an intelligent control system where the IT2-FLC is the main controller. We will designed a SIT2-FLC for the game Flappy Bird while a DIT2-FLC structure for the game moon lander. We examined the performance of both IT2 fuzzy control systems with respect to their control system and game performances in comparison with its T1 and conventional counterparts. Thereby, we have shown that the resulting IT2-FLCs resulted with an adequate control and game performance in the presence of the uncertainties, disturbances and nonlinear system dynamics.

References

[1] S. M. Lucas and G. Kendall, "Evolutionary computation and games," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 10-18, 2006.

[2] N. Cole, S. J. Louis and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," in *Proc. IEEE Congr. on Evol. Comput.*, 2004, pp.139-145.

[3] J. MacGlashan and M. L. Littman, "Between imitation and intention learning," in *Proc. Int. Conf. on Artificial Intell.*, 2015, pp. 3692-3698.

[4] P. Hingston, "A turing test for computer game bots," *IEEE Trans. Computational Intell. and AI in Games*, vol. 1, no. 3, pp.169-186, 2009.

[5] S. M. Lucas, "Evolving a neural network location evaluator to play ms. pac-man," in *Proc. IEEE Symp. on Comput. Intell. and Games*, 2005, pp. 203–210.

[6] B. Sheppard, "World-championship-caliber Scrabble," *Artificial Intelligence*, vol. 134, no. 1, pp. 241-275, 2002.

[7] D.Perez, G. Recio, Y. Saez and P. Isasi, “Evolving a fuzzy controller for a car racing competition,” in *Proc. IEEE Symposium on Computational Intelligence and Games*, September 2009, pp. 263-270.

[8] W. Goufang, F. Zhou, L. Ping and L. Bo, “Shaping in reinforcement learning via knowledge transferred from human demonstrations,” in *Proc. 34th Chinese Control Conference,* July 2015, Hangzhou, China, pp. 3033-3038.

[9] S. Karakovskiy and J. Togelius, “The mario ai benchmark and competitions,” *IEEE Trans. Comput. Intell. and AI in Games*, vol. 4, no. 1, pp. 55-67, 2012.

[10] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill and M. Preuss, “A survey of real-time strategy game ai research and competition in starcraft,” *IEEE Trans. on Comput. Intell. and AI in games*, vol. 5’ no. 4, pp. 293-311, 2013.

[11] B. Takacs, J. Holaza, J. Stevek and M. Kvasnica, “Export of explicit model predictive control to python,” in *Proc. 20th International Conference on Process Control,* June 2015, Štrbské Pleso, Slovakia, pp. 78-83.

[12] A. Sahin, E. Atici and T. Kumbasar, “Type-2 fuzzified flappy bird control system,” in *Proc. IEEE Int. Conf. on Fuzzy Syst.,* 2016, pp.1578-1583.

[13] M. Ebner, J. Levine, S. M. Lucas, T. Schaul, T. Thompson and J. Togelius, “Towards a video game description language,” DOI: 10.4230/DFU.Vol6.12191.85, 2013.

[14] A. Sahin and T. Kumbasar, “Landing on the Moon with Type-2 Fuzzy Logic,” in *Proc. IEEE Int. Conf. on Fuzzy Syst.,* 2017, *Accepted*.

[15] J. M. Mendel and R. B. John, “ Type-2 fuzzy sets made simple,” *IEEE Trans. on Fuzzy Syst.*, vol. 10, no. 2, pp. 117-127, 2002.

[16] J. M. Mendel, H. Hagras, W.W Tan, W.W. Melek, and H. Ying, “Introduction to type-2 fuzzy logic control”, John Wiley and IEEE Press, Hoboken, NJ, 2014.

[17] H. Hagras, “A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots,” *IEEE Trans. Fuzzy Syst.,* vol. 12, no. 4, pp. 524-539, 2004.

[18] T. Kumbasar and H. Hagras, H. “Big Bang–Big Crunch optimization based interval type-2 fuzzy PID cascade controller design strategy,” *Inform. Sci.*, vol. 282, pp. 277-295, 2014.

[19] A. Kumar and V. Kumar, “Evolving an interval type-2 fuzzy PID controller for redundant robotic manipulator,” *Expert Systems with Applications,* vol. 73, pp. 161-177, 2017.

[20] T. Kumbasar and H. Hagras, “A self-tuning zSlices based general type-2 fuzzy PI controller,” *IEEE Trans. Fuzzy Syst*., vol. 23, no. 4, pp. 991-1013, 2015.

[21] M. Mehndiratta, E. Kayacan and T.Kumbasar, “Design and experimental validation of single input type-2 fuzzy PID controllers as applied to 3 DOF helicopter testbed,” in *Proc. IEEE Int. Conf. on Fuzzy Syst.*, 2016, pp. 1584-1591.

[22] C. Lynch, H. Hagras and V. Callaghan, “Embedded type-2 FLC for real-time speed control of marine and traction diesel engines,” in *Proc. IEEE Int. Conf. on Fuzzy Syst.,* 2005, pp.347-352.

[23] E. Yesil, “Interval type-2 fuzzy PID load frequency controller using Big Bang–Big Crunch optimization, ” *Applied Soft Computing*, vol. 15, pp. 100-112, 2014.

[24] Q. Liang and J. M. Mendel, “Interval type-2 fuzzy logic systems: theory and design,” *IEEE Trans. Fuzzy Syst*., vol. 8, no. 5, pp. 535-550, 2000.

[25] H. Wu and J. M. Mendel, “Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems,” *IEEE Trans. Fuzzy Syst*., vol. 10, no. 5, pp. 622-639, 2002.

[26] T. Kumbasar, “ A simple design method for interval type-2 fuzzy PID controllers. Soft Computing,” Soft Computing, vol. 18, no. 7, pp. 1293-1304, 2014.

[27] T. Kumbasar, “Robust stability analysis and systematic design of single input interval type-2 fuzzy logic controllers,” *IEEE Trans. Fuzzy Syst.*, 10.1109/TFUZZ.2015.2471805, 2015.

[28] A. Taskin and T. Kumbasar, “An open source Matlab/Simulink toolbox for interval type-2 fuzzy logic systems,” in *Proc. IEEE Symp. Series on Comput. Intell.,* 2015, pp. 1561-1568.

[29] T. Kumbasar and H. Hagras, Interval type-2 fuzzy PID controllers, In *Springer Handbook of Computational Intelligence,* Springer Berlin Heidelberg, pp. 285-294, 2015.

[30] A. Sakalli, T. Kumbasar, M. F. Dodurka and E. Yesil, “The simplest Interval Type-2 Fuzzy PID Controller: structural analysis,” in *Proc. IEEE Int. Conf. on Fuzzy Syst.,* 2014, pp. 626-633.

[31] A. Sakalli and T. Kumbasar, “On the fundamental differences between the NT and the KM center of sets calculation methods on the IT2-FLC performance” in *Proc. IEEE Int. Conf. on Fuzzy Syst.,* 2015, pp. 1-8.

[32] A. Sakalli, A. Bekea and T. Kumbasar, “Gradient Descent and Extended Kalman Filter based Self-Tuning Interval Type-2 Fuzzy PID Controllers” in *Proc. IEEE Int. Conf. on Fuzzy Syst.,* 2016, pp. 1592-1598.

[33] A. Isaksen, D. Gopstein and A. Nealen, “Exploring game space using survival analysis,” *Foundations of Digital Games,* 2015.

[34] M. Zhang. (2015, October 25). *Flappy bird for MATLAB* [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/4579 5-flappy-bird-for-matlab.

[35] W.-C. So, C. K. Tse, and Y.-S. Lee, “Development of a fuzzy logic controller for dc–dc converters: Design, computer simulation, and experimental evaluation,” *IEEE Trans. Power Electron.,* vol. 11, no. 1, pp. 24-32, 1996.

[36] B. Edwards, "Forty Years of Lunar Lander", Technologizer, 2009. [Online]. Available: http://www.technologizer.com/2009/07/19/lunar-lander/. [Accessed: 02- Jan- 2017].

[37] S. Samothrakis, S. A. Roberts, D. Perez and S. M. Lucas, “Rolling horizon methods for games with continuous states and actions,” in *Proc. IEEE Conf. on Comput. Intell. in Games*, 2014, pp. 1-8.

[38] S. A. Roberts and S. M. Lucas, “Measuring interestingness of continuous game problems,” in *Proc. IEEE Conf. on Comput. Intell. in Games*, 2013, pp. 1-8.

[39] H. Corte, "Moon Lander matlab game", mathworks.com, 2012. [Online]. Available: https://uk.mathworks.com/matlabcentral/fileexchange/38927. [Accessed: 02- Jan- 2017].